Corso Architettura degli elaboratori II – Prof. Borghese – Primo Compitino, 11.Aprile.2022, v1

Cognome e nome: ***Matricola:***

1) [7] Data la CPU N. 1, specificare il contenuto di **tutte le linee** (dati e controllo), quando è in esecuzione il seguente segmento di codice:

```
0x00000400 and $s5, $t2, $t1
0x00000404 lw $s1, 8($s0)
0x00000408 or $t4, $s5, $s1
0x0000040C addi $t1, $s1, 100
0x00000410 sw $s2, 32($t1)
0x00000414 sub $s2, $s0, $s2
```

quando l'istruzione di and si trova in fase di WB. **Sottolineare quali linee trasportano segnali utili**.

2. [7] Identificare le dipendenza e gli hazard su questo segmento di codice.

```
0x00000400 and $s5, $t2, $t1
0x00000404 addi $s1, $s5, 32
0x00000408 or $t4, $s6, $s1
0x0000040C sw $s5, 32($s0)
0x00000410 lw $s5, 32($s1)
0x00000414 sw $s5, 32($s2)
0x00000418 or $t4, $s6, $s1
0x0000041C beq $t4, $t5, 24
```

E modificare la CPU N.2, dove e' possibile, per evitare stalli e gestire correttamente gli hazard.

3. [6] Cosa sono gli interrupt e le eccezioni? Come vengono gestiti dalle architetture Intel e dalle architeture MIPS/ARM? Specificare gli elementi della CPU MIPS che sono dedicati alla gestione delle eccezioni e cosa contengono. Modificare la CPU in Figura N.3 per potere gestire un'eccezione di "Overflow". Cosa si intende per mascheramento degli interrupt? Viene praticato nei MIPS? Come vengono gestite le eccezioni e gli interrupt dai sistemi operativi sul MIPS? Scrivere uno scheletro di funzione assembler per gestire un'eccezione.

4. [8] Modificare la pipeline in Figura 3 perché diventi una pipeline superscalare. Spiegare la ragione e lo scopo di **tutte le modifiche** più rilevanti da apportare ai diversi stadi. Che differenza c'è tra pipeline super-scalare e pipeline dotata di VLIW? Quali sono i vantaggi e gli svantaggi di un approccio rispetto all'altro. Qual è il migliore e perché? Descrivere come funzionano le seguenti tecniche e dire se sono tecniche principalmente **software** o **hardware** e perchè. In alcuni casi la risposta corretta può essere entrambi gli approcci. Identificare quali sono i **punti forti** ed i **punti deboli**.

a) Predizione dei salti
b) Branch predicion buffer
c) Speculazione
d) Parallelizzazione dell'esecuzione
e) Pipeline superscalari
f) Esecuzione fuori ordine
g) Reservation station
h) Buffer di riordino
i) Ridenominazione dei registri
j) Issue

5. [2] Come si implementa l'esecuzione vettoriale in una pipeline super-scalare? Mostrarlo modificando un cammino di esecuzione di una pipeline non dotata di esecuzione vettoriale.

6. [2] Descrivere come funzionano le seguenti tecniche e dire se sono tecniche principalmente **software** o **hardware** e perchè. In alcuni casi la risposta corretta può essere entrambi gli approcci. Identificare quali sono i **punti forti** ed i **punti deboli**.

a) Superpipeline

b) Branch delay slot
c) Hazard
d) Bolla
e) Stallo

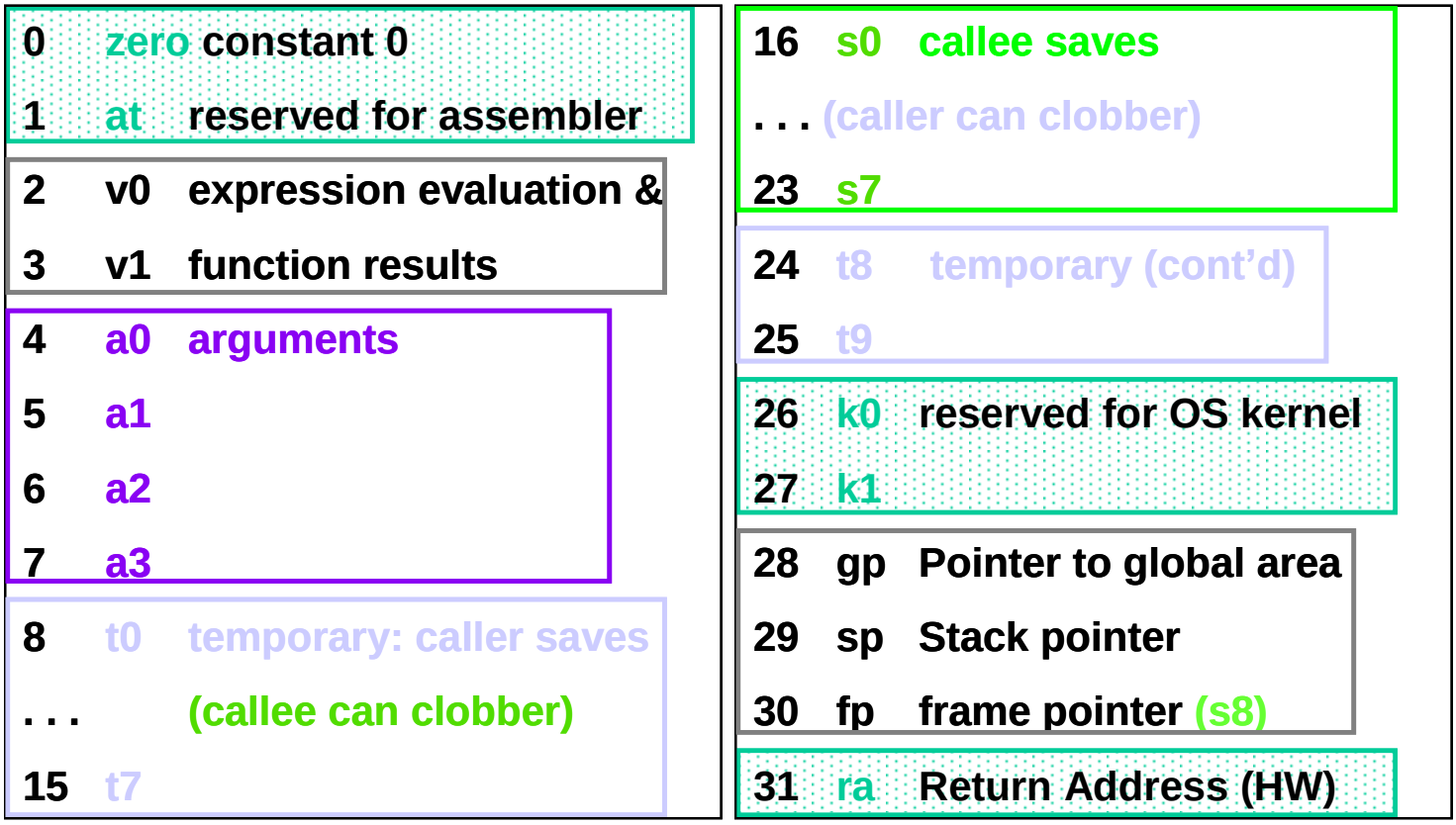
7. [4] Riscrivere il codice seguente in modo che sia eseguito nel minor tempo possibile su una pipeline con 4 cammini di esecuzione di cui 3 general purpose (in grado di eseguire tutte le istruzioni) e 1 dedicato solo alle istruzioni di memoria. Qual è lo speed-up? (rapporto tra le prestazioni prima della modifica, sul codice sequenziale, e dopo la modifica). Applicare lo srotolamento dei cicli per un numero massimo di 8 iterazioni del ciclo. Si supponga di avere a disposizione un numero di registri interni di pipeline sufficientemente grande e che si possa applicare la ridenominazione dei registri.

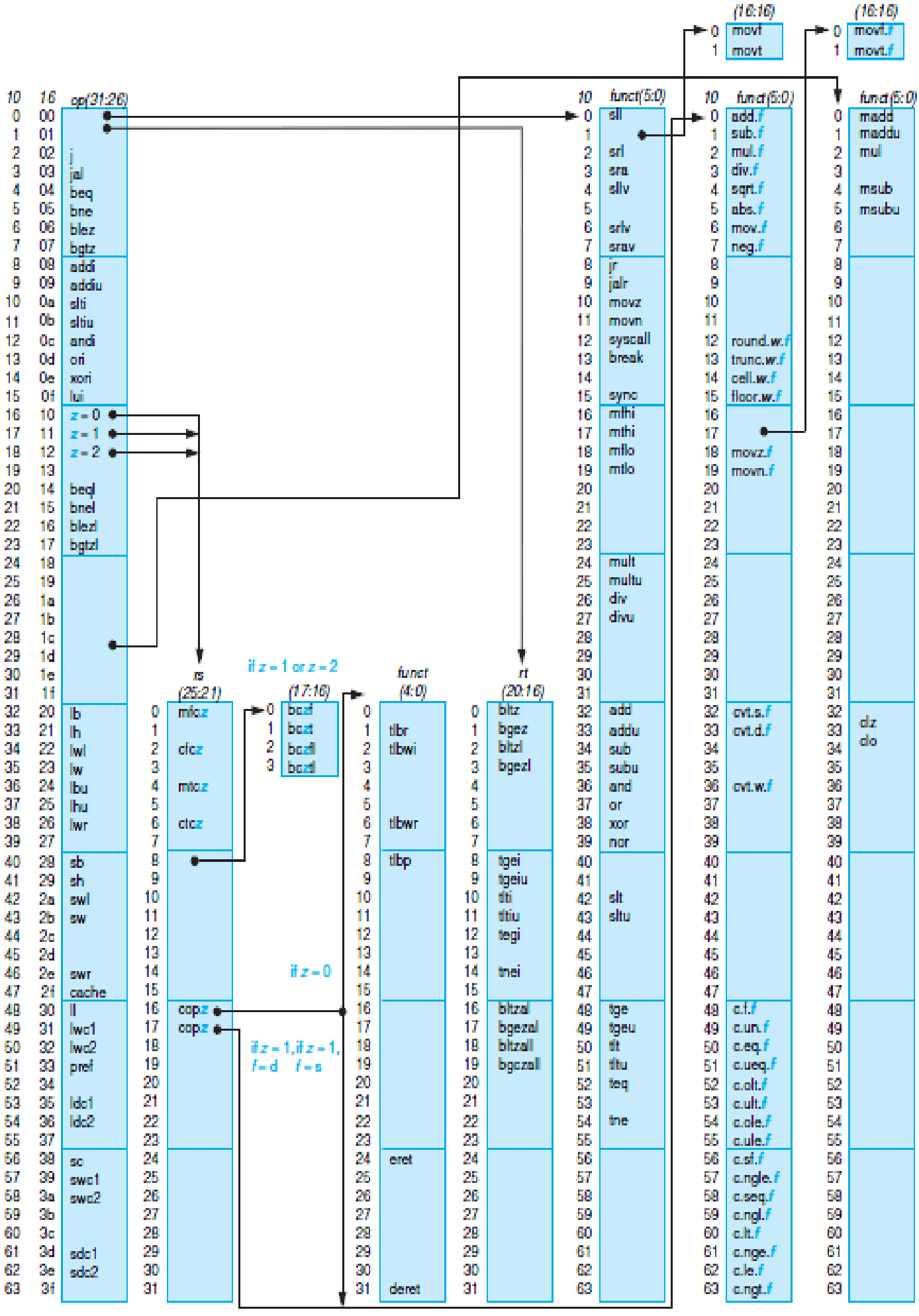
```
Ciclo:    lw   $t0, 0($s1)          # M[s1] -> t0
          addu $t0, $t0, $s2        # t0 = t0 + s2
          sw   $t0, 0($s1)          # M[s1] <- t0
          addi $s1, $s1, -4         # next element
          bne  $s1, $zero, Ciclo
          or   $s6, $s7, $s5
```

Il codice assembler corrisponde al seguente codice C:

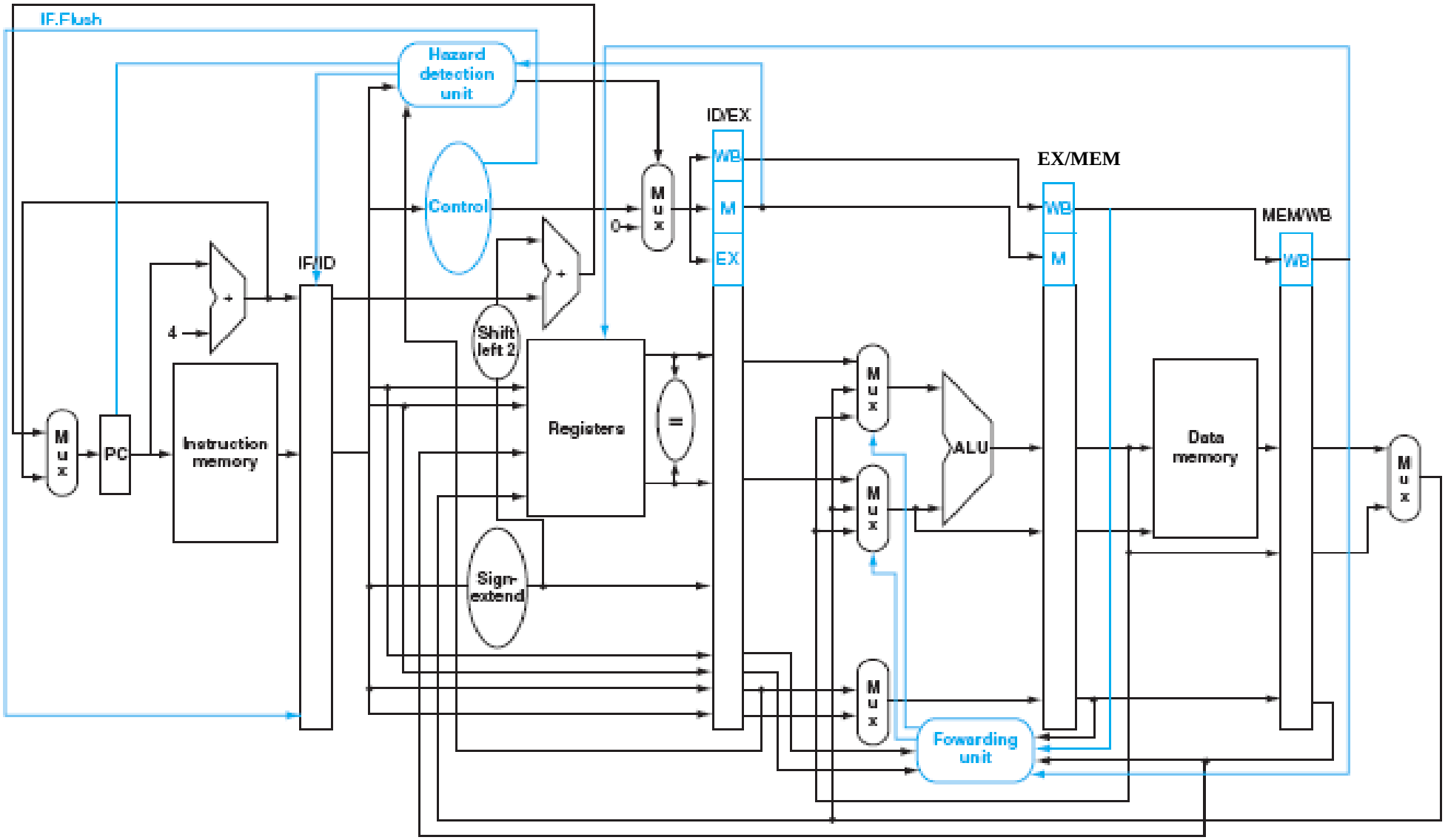
```
Ciclo:    v[s1] = t0 + v[s1];
          s1--;
          if (s1 != 0) goto Ciclo
```

Registri del register file

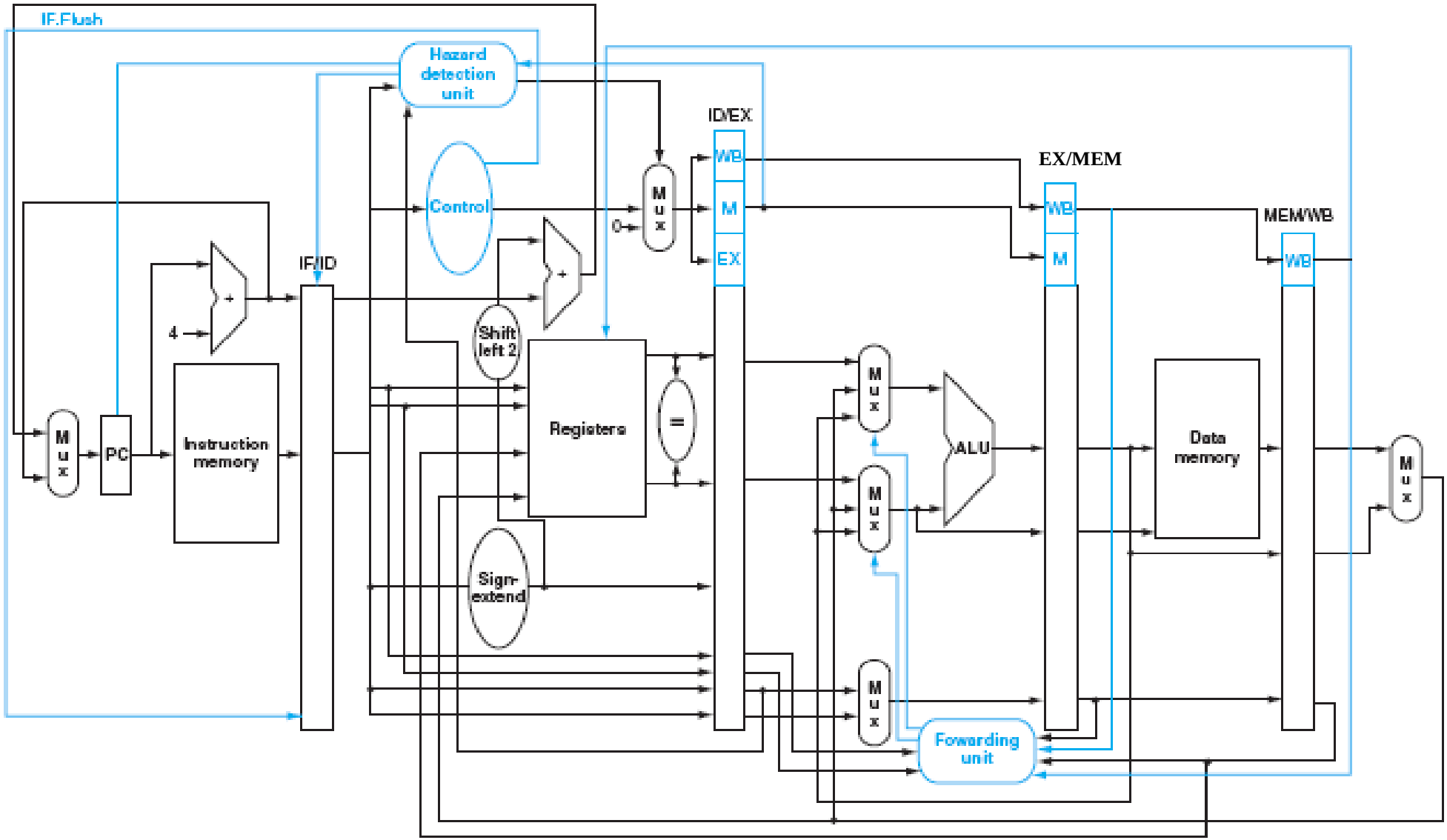



op(31:26)
j
jal
beq
bne
blez
bgtz
addi
addiu
slti
sltiu
andi
ori
xori
lui
z = 0
z = 1
z = 2
beql
bnel
blezl
bgtzl
lb
lh
lwl
lw
lbu
lhu
lwr
sb
sh
swl
sw
swr
cache
ll
lwc1
lwc2
pref
ldc1
ldc2
sc
swc1
swc2
sdc1
sdc2
rs (25:21)
mfcz
cfcz
mtcz
ctcz
copz
copz
if z = 1 or z = 2
(17:16)
bczf
bczt
bczfl
bczfl
if z = 0
if z = 1, if z = 1,
f = d f = s
funct (4:0)
tlbr
tlbwi
tlbwr
tlbp
eret
deret
rt (20:16)
bltz
bgez
bltzl
bgezl
tgei
tgeiu
tlti
tltiu
teqi
tnei
bltzal
bgezal
bltzall
bgezall
funct(5:0)
sll
srl
sra
sllv
srlv
srav
jr
jalr
movz
movn
syscall
break
sync
mfhi
mthi
mflo
mtlo
mult
multu
div
divu
add
addu
sub
subu
and
or
xor
nor
slt
sltu
tge
tgeu
tlt
tltu
teq
tne
(16:16)
movf
movt
(16:16)
movf.f
movt.f
funct(5:0)
add.f
sub.f
mul.f
div.f
sqrt.f
abs.f
mov.f
neg.f
round.w.f
trunc.w.f
ceil.w.f
floor.w.f
movz.f
movn.f
cvt.s.f
cvt.d.f
cvt.w.f
c.f.f
c.un.f
c.eq.f
c.ueq.f
c.olt.f
c.ult.f
c.ole.f
c.ule.f
c.sf.f
c.ngle.f
c.seq.f
c.ngl.f
c.lt.f
c.nge.f
c.le.f
c.ngt.f
funct(5:0)
madd
maddu
mul
msub
msubu
clz
clo

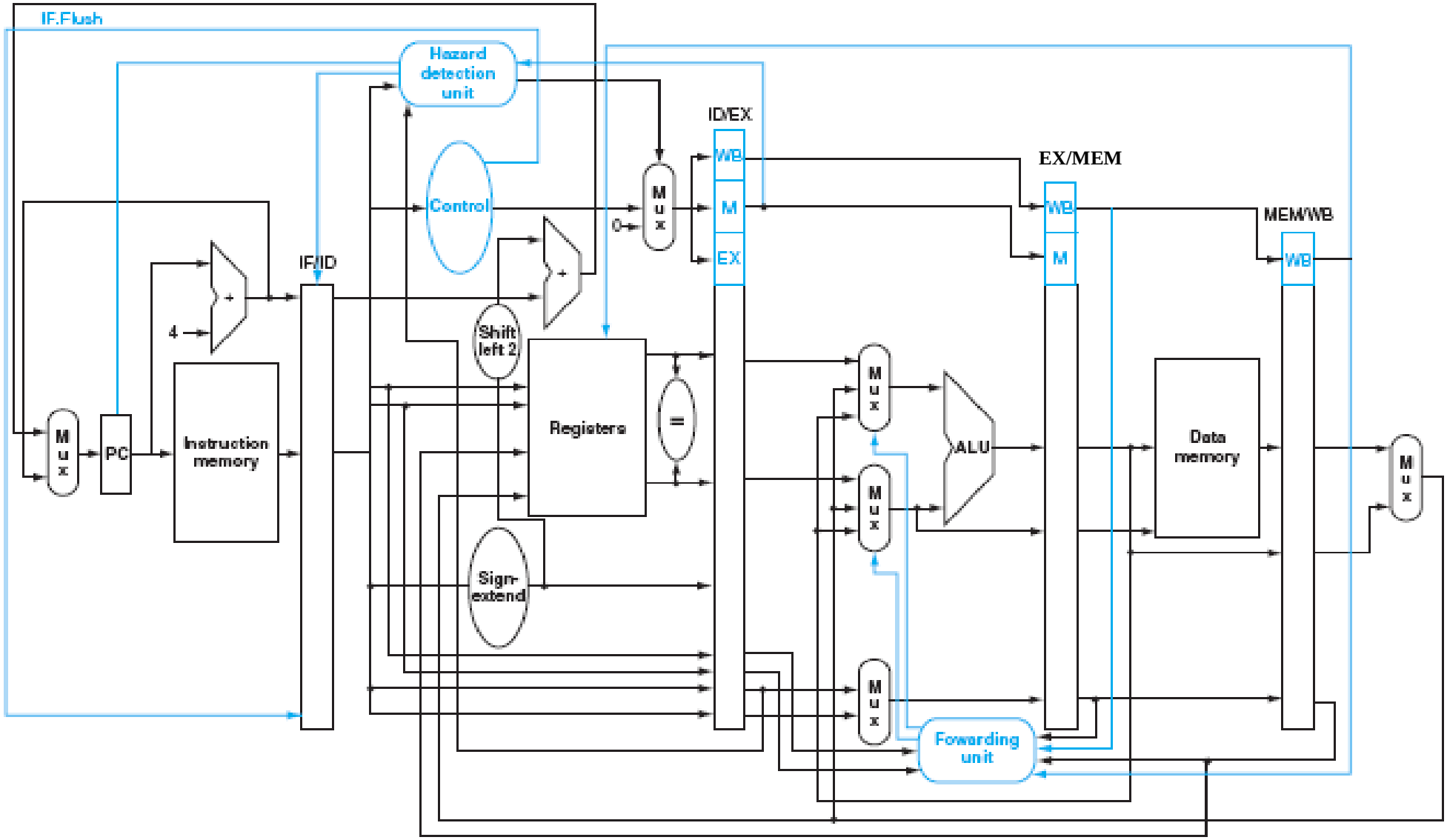
CPU N. 1

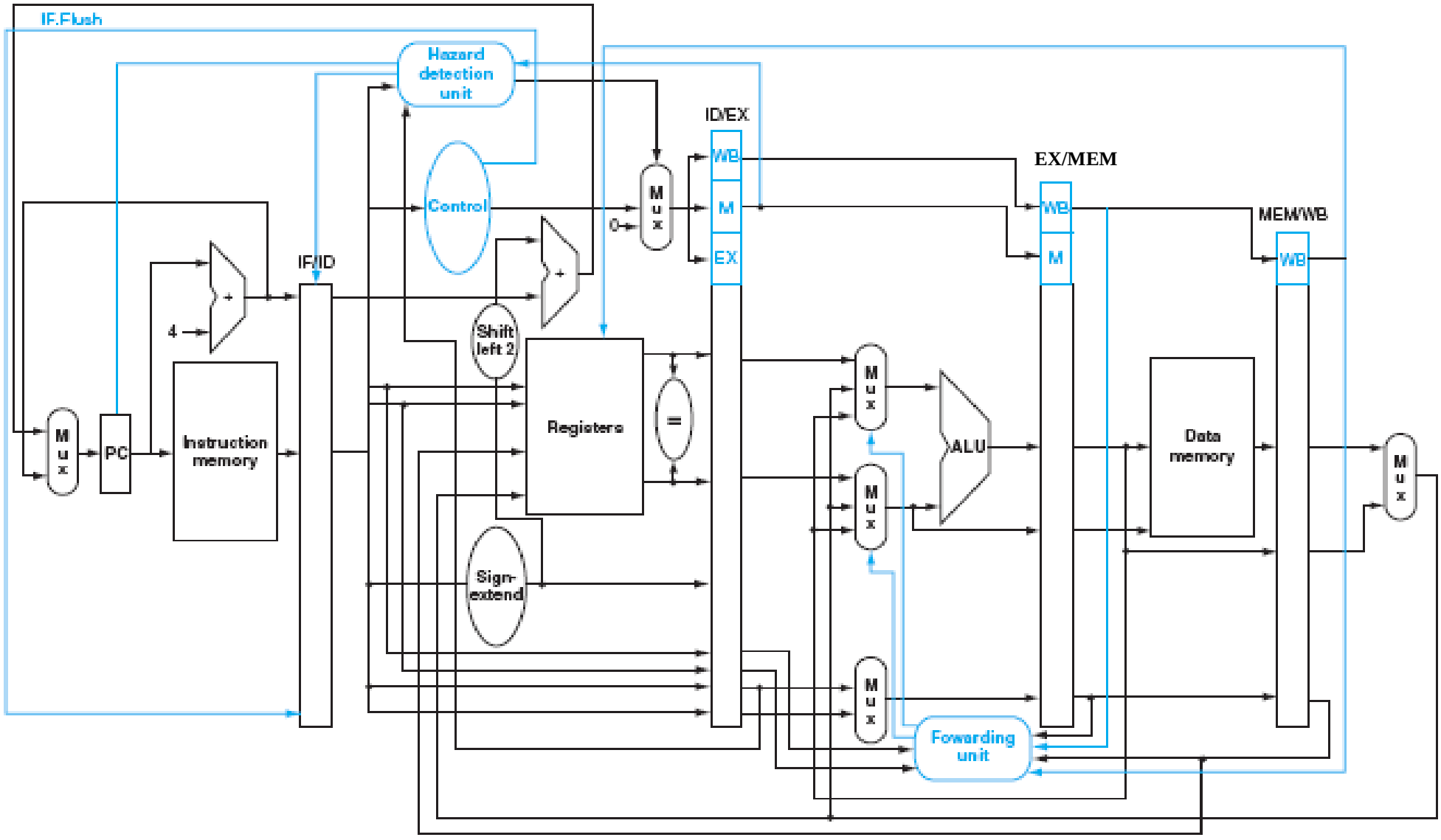

IF.Flush
Hazard detection unit
Control
ID/EX
WB
M
EX
EX/MEM
WB
M
MEM/WB
WB
M u x
0
IF/ID
+
4
Shift left 2
Registers
=
M u x
PC
Instruction memory
Sign-extend
ALU
Data memory
Fowarding unit

CPU N. 2


IF.Flush
Hazard detection unit
Control
ID/EX
WB
M
EX
EX/MEM
WB
M
MEM/WB
WB
IF/ID
0
M u x
4
+
Shift left 2
Registers
=
ALU
Data memory
PC
Instruction memory
Sign-extend
Fowarding unit

CPU N. 3


IF.Flush
Hazard detection unit
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
0
M u x
IF/ID
+
4
Shift left 2
Registers
=
PC
Instruction memory
Sign-extend
ALU
Data memory
Fowarding unit


IF.Flush
Hazard detection unit
Control
ID/EX
WB
M
EX
EX/MEM
MEM/WB
0
M u x
IF/ID
4
+
Shift left 2
PC
Instruction memory
Registers
=
ALU
Data memory
Sign-extend
Fowarding unit